Programmation Web « Client Riche » M413 - TD n°4 (séances 5 & 6)

1 Objectifs

Ce TD illustre la partie du cours sur la programmation web et en particulier les concepts liés à la conception d'application riches coté client, c'est-à-dire dans votre navigateur.

Les principaux concepts abordés seront :

- Les principes avancés du langage JavaScript.
- L'exploitation du DOM.
- L'Ansychronisme en JavaScript : AJAX, XML et JSON.
- Mise en œuvre d'une application en JavaScript.

Vos « **livrables** » qui seront à déposer dans **la boîte de dépôt ad-hoc sur le LMS Moodle d'UCA avant la date spécifiée**, seront basés sur l'arborescence fournie et constitués **a minima** des **dossiers assets/, css/, js/** et des **fichiers** suivants :

- Les réponses à chacune des questions posées devrons être rédigées au format « Markdown » dans le fichier « **README.md** » fournie.
- Des fichiers index.html, *.html, js/td{x}.js js/*.js, css/td{x}.css, css/*.css et plus si nécessaire…

https://fr.wikipedia.org/wiki/Markdown

2 Documentation

Pour ce TD, mais également pour les suivants, sauf mention contraire dans l'énoncé ou si un lien vous est fourni, vous ne devez pas aller chercher des réponses sur internet !

Seuls les sites suivants sont autorisés :

- Le cours sur : https://lms.univ-cotedazur.fr/
- Le site de Mozilla : https://developer.mozilla.org/fr/
- Les sites du W3C : pour la validation HTML5 ou du CSS
  Mais aussi pour la documentation sur le DOM.

3 Prise en main de l'environnement

Pour réaliser ce TD, vous aurez également besoin d'un éditeur de texte pour écrire le code de vos pages. Vous pouvez par exemple utiliser un des logiciels **Brackets**, **Notepad**++, **Eclipse**, etc.

Pensez à sauvegarder régulièrement votre travail et à commenter votre code.

N'hésitez pas à sauvegarder les différentes versions, évolutions d'un même exercice de manière à pouvoir facilement réviser ou le réutiliser par la suite…

4 Les Outils pour parcourir le Document Object Model

De nos jours, des outils de débogage évolués sont intégrés aux différents navigateurs (cf. cours). Ces outils vous seront très utiles pour explorer l'arbre DOM de votre page, vérifier les propriétés CSS ou encore connaitre les erreurs de votre code JavaScript !

5 Rappel de cours.

Lors de ce TD, l'ensemble des pages web écrites devra (sauf mention contraire) être constitué de pages XHTML5 valide ne contenant pas de mise en forme (squelette index.html et *.html fournis).

La mise en forme sera dans un ou plusieurs fichiers CSS valides
(squelettes css/td{x}.css css/*.css fournis).

De même, le code JavaScript sera également dans des fichiers séparés
(squelettes js/td{x}.js js/*.js fournis).

Bien lire l'ensemble des consignes avant de commencer à coder.

6 Exercice 1 : 2048 (obligatoire).

Si vous n'avez pas fini l'exercice 2 du TD précédent, vous avez **1h30 maximum** pour le terminer !

Vous devrez finir les exercices obligatoires du TD précédent chez vous et les déposer dans la boite de dépôt ad-hoc avant la date indiquée.

Attention, la qualité du code produit pour l'exercice sur le 2048 sera évaluée.

Aussi, pensez à nommer vos variables convenablement (en anglais de préférence), commenter (en anglais de préférence) et bien indenter votre code.

L'ensemble des constantes devra être regroupé en début de fichier pour être facilement modifié en cas de besoin. On ne retrouvera donc aucune valeur de constante dans le code…

Vous pouvez ensuite passer à l'exercice suivant de ce TD.

7 Exercice 2 : RSS Reader (obligatoire).

RSS (Really Simple Syndication) est une famille de formats de données utilisés pour la « syndication » (i.e. souscription) de contenus Web.

Un produit RSS est une ressource du World Wide Web dont le contenu est produit généralement automatiquement en fonction des mises à jour d'un site Web.

Les flux RSS sont des fichiers XML qui sont souvent utilisés par les sites d'actualité et les blogs pour présenter les titres des dernières informations consultables.

Pour plus de détails sur les flux RSS : https://fr.wikipedia.org/wiki/RSS

7.1 Mise en place basique du protocole de communication asynchrone.

Commencez par compléter la page web « **rss.html** » (XHTML5 & CSS) fournie qui devra proposer de choisir un flux dans une liste.

Pour le moment la liste sera composée de trois noms. Chaque nom sera associé à l'une des URLs suivantes :

- La SIF : https://www.societe-informatique-de-france.fr/feed/
- L'UCA : https://univ-cotedazur.fr/adminsite/webservices/export_rss.jsp?NOMBRE=10&CODE_RUBRIQUE=ACCUEIL-FR&LANGUE=0
- Les actualités de l'« Informaticien » : https://linformaticien.com/feed/

Cette liste pourra prendre différentes formes (**<select>**, **
**, …), à vous d'être créatif ;)

Une zone de type ** ** pourra être ajoutée pour faciliter les insertions à venir.

Ajoutez à votre fichier de script les fonctions suivantes :

- **init()** : au chargement de la page elle crée un objet de type **XMLHttpRequest** accessible à toutes les autres fonctions.
- **selectRSS(value)** : quand l'utilisateur sélectionne un flux RSS dans la liste, on s'abonne au changement d'état de l'objet **XMLHttpRequest** : lorsque l'on aura reçu le document, on appellera la fonction de traitement **myCallback()**. Il faudra ensuite préparer la requête HTTP asynchrone de type **GET** et l'envoyer.
- **myCallback(data)** : insère simplement les données de type texte dans la page HTML.

> - Quelle erreur obtenez-vous ?
> - Pourquoi la solution ci-dessus de fonctionne pas ?

https://fr.wikipedia.org/wiki/XMLHttpRequest

https://developer.mozilla.org/fr/docs/Web/Guide/AJAX/Premiers_pas

https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest/Utiliser_XMLHttpRequest

https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest

7.2 Les fichiers locaux.

Pour résoudre ce problème nous allons commencer par mettre en place une première solution simple :

- Enregistrez une copie locale des trois fichiers XML précédent.
- Modifiez votre code pour lire les fichiers locaux : les fonctions modifiées seront suffixées par un 2 (exemple **init2()**), de manière à conserver les fonctions de chaque version.
- Vérifiez que maintenant tout fonctionne correctement.

7.3 Un « Proxy » en PHP.

La solution précédente nous a permis de valider le fonctionnement de notre code, mais celle-ci n'est pas satisfaisante.

Une solution, pour ne pas faire de « cross-domaine » est d'utiliser un petit proxy en PHP.

Vous trouverez dans le fichier « **proxyRSS-XML.php** » le code du proxy que nous allons utiliser pour lire les fichiers RSS et les transmettre à notre client web.

*Attention : vous n'avez pas le droit de modifiez ce fichier « **proxyRSS-XML.php** » !*

- Modifiez votre code pour qu'il appelle le proxy en lui passant en paramètre l'URL du flux recherché. Les fonctions modifiées seront suffixées par un 3 (exemple **init3()**), de manière à conserver les fonctions de chaque version.
- Vérifiez que tout fonctionne toujours correctement.

Le **script du proxy en PHP** devra être mis en œuvre dans un dossier **www/M413/** de votre compte Unix sur le serveur « **linserv-info-03.campus.unice.fr** ».

Ce serveur est accessible en dehors de l'IUT via le VPN de étudiant de NICE que vous avez déjà utilisé pour le module M314…

7.4 Analyse du contenu et mise en page.

Puisque nous recevons un fichier RSS, il y a probablement mieux à faire que d'afficher simplement du texte.

En effet, nous savons qu'il s'agit d'un fichier XML. De plus, nous savons qu'il respecte la DTD décrivant les flux RSS.

Nous allons donc mettre en place un script capable d'analyser ces documents et de nous les présenter de manière plus agréable dans notre page.

Pour cela nous nous reposerons sur la description des balises présente sur cette page :

https://fr.wikipedia.org/wiki/RSS#Explication_des_principales_balises

Pour cela, vous devrez :

- Ajouter à votre page une présentation des métadonnées suivantes (si celles-ci sont disponibles) : titre, description, lien et date de publication. A loisirs vous pouvez également ajouter les 4 autres métadonnées.
- Ajouter à votre page un affichage dédié pour chaque article. On devra avoir un affichage :
    - simple mais compact présentant qu'une partie des informations.
    - un affichage détaillé permettant d'avoir accès à l'intégralité des données contenus dans le flux rss.
- Augmenter la liste de liens RSS disponibles 2 ou 3 autres liens de votre choix.
- Enfin, ajouter un champ texte permetant à l'utilisateur de saisir l'url du flux RSS si celui-ci n'est pas disponible dans votre liste.

7.5 Données en JSON.

Pour tester l'exploitation des données en JSON, nous allons simplement demander à notre proxy de convertir les données dans ce format avant de nous les transmettre :

Modifiez votre code pour qu'il utilise le fichier « **proxyRSS-JSON.php** » fourni.

Modifiez le traitement des données qui sont maintenant en JSON (texte) et non plus en XML.

Le script **proxyRSS-JSON.php** devra également être mis en œuvre dans le dossier **www/M413/** de votre compte Unix sur le serveur « **linserv-info-03.campus.unice.fr** ».

7.6 Un concurrent : Atom (conseillé).

Un document au format Atom est appelé un « fil de syndication Atom » ou fil Web. Ces fils peuvent être affichés aussi bien sur un site Web que directement dans un agrégateur, qui est un logiciel prévu à cet effet. Ainsi, cela permet de suivre, ou de « s'abonner », à un fil.

Le propriétaire d'un site web peut quant à lui utiliser un logiciel spécialisé, tel qu'un système de gestion de contenu, pour publier une liste de ressources, dans un format standardisé et lisible par une machine, et dont il souhaite notifier des mises à jour.

Le développement d'Atom a été justifié par le manque de flexibilité commun aux nombreuses variantes de RSS.

Pour cette exercice, vous devrez :

Modifier votre code pourqu'il supporte en même temps les documents de type Atom 1.0 et RSS 2.0.

Les correspondances entre les balises sont disponibles à cette URL : https://fr.wikipedia.org/wiki/Atom_Syndication_Format.